# オープンソースソフトウェアに対するSRGMに基づく予測精度の検証 (不確実性と意思決定の数理)

AUTHOR(S):

田村, 慶信; 田中, 智朗; 山田, 茂

CITATION:

田村, 慶信 ...[et al]. オープンソースソフトウェアに対するSRGMに基づく予測精度の検証 (不確実性と意思決定の数理). 数理解析研究所講究録 2009, 1636: 243-250

ISSUE DATE:

2009-04

URL:

http://hdl.handle.net/2433/140475

数理解析研究所講究録
第1636巻 2009年 243-250



# オープンソースソフトウェアに対するSRGMに基づく予測精度の検証

広島工業大学・情報学部 田村 慶信 (Yoshinobu Tamura) †
†Faculty of Applied Information Science, Hiroshima Institute of Technology
鳥取大学大学院・工学研究科社会基盤工学専攻 田中 智朗 (Tomoaki Tanaka) ‡
鳥取大学大学院・工学研究科 山田 茂 (Shigeru Yamada) ‡
‡Graduate School of Engineering, Tottori University

## 1 はじめに

現在のソフトウェア開発は，多くの開発者が協調しながら開発を行うオープンソースプロジェクトなどの様々な分散開発形態が存在する．特に，ネットワーク環境を利用して開発されるオープンソースソフトウェア (Open Source Software，以下OSSと略す) は，世界中の誰もが開発に参加でき，ソースコードが公開され，誰でも自由に改変可能なソフトウェアであることから，組込みシステムやサーバ用途として広く採用され，急激に普及が広まっている [1]．また，オープン規格やOSSを利用することによって，電子行政機関がプライバシーや個人の自由を保護するとともに，市民が電子政府と情報をやり取りできるようにするのに役立つことから，EU加盟国を中心に欧米においても政府関係機関がOSSを支持する動きが広がっている [2]．

一方，OSSの利用に関しては，未だに多くの不安が残されている．まず第1に，システム導入後のサポートや品質上の問題といった利用者側の一般的な不安である．第2に，OSSは本当にビジネスになるのか，オープンソースのソフトウェアを事業化することによって自社製のソフトウェア商品までが市場を失うことにならないか，といった開発者側の不安である [2]．特にサポートや品質上の問題については，OSSの普及を妨げる大きな要因として考えられている．本論文では，こうしたオープンソースプロジェクトの下で開発されているOSSに対する信頼性評価法を提案するとともに，実際に公開されているOSSに対する信頼性評価法の適用可能性について考察する．

従来から，ソフトウェア製品の開発プロセスにおけるテスト進捗管理や出荷品質の把握のための信頼性評価を行うアプローチとして，ソフトウェア故障の発生現象を不確定事象として捉えて確率・統計論的に取り扱う方法がとられている．その1つが，ソフトウェア信頼度成長モデル (Software Reliability Growth Model，以下SRGMと略す) である [3]．

また，OSSに対する現在の研究動向としては，設計工程や開発手法を対象とした文献はいくつか提案されているが [4, 5]，動的解析に基づいた信頼性評価に関する研究はほとんど行われていないのが現状である．さらに，OSSに対して傾向分析に基づく事例研究としての文献がいくつか提案されているが，これらはOSSのもつ特有の開発形態を考慮した信頼性評価手法を提案したものではない [6, 7]．

本論文では，こうしたOSSの特殊な開発形態を包括した信頼性評価法として，既存の非同次ポアソン過程および確率微分方程式に基づいたソフトウェア信頼度成長モデルの適用可能性について考察する．また，実際のオープンソースプロジェクトにおけるバグトラッキングシステムから採取されたフォールトデータに対する数値例を示すとともに，各モデルを適用した場合における予測精度について議論する．

## 2 各コンポーネントに対する信頼性評価

ソフトウェアの信頼性評価手法の開発において，各コンポーネントでのデバッグの状況やその良し悪しが，システム全体の信頼性に与える影響を考慮しようとする場合，プログラムパス，コンポーネントの規模，フォールト報告者のスキルなどの，様々に絡み合った要因を捉える必要があると考えられる．こうした複雑な状況下でシステム全体の信頼性に対する各コンポーネントの影響度合いを推定するために，本論文ではニューラルネットワー

ク [8] を適用する．これにより，バグトラッキングシステム上から採取されたデータのみに基づいて機械的に各コンポーネントのシステム全体に与える重要度を推定することが可能となる．

OSS において，システム全体の信頼性に対する各コンポーネントの影響を考えた場合，コンポーネントの規模，フォールト報告者のスキル，フォールト修正の状態，コンポーネントの開発時間，コンポーネント間のパスの数，コンポーネント間の入出力データ量といった様々な要因を考慮する必要がある．こうした複雑な状態を適当な仮定の下で物理的な意味からモデル化することは困難である．したがって，本論文では，各コンポーネント間の内部状態をブラックボックスとして捉えるためにニューラルネットワークを適用する．すなわち，入力と出力の関係から，その内部構造をニューラルネットワークにより学習させることによって，各コンポーネントがシステム全体の信頼性に与える影響度合いを推定する．これにより，バグトラッキングシステム上から採取されたデータのみに基づいた信頼性評価が可能となることから，実利用上においても容易に適用できるものと考える．本論文においては簡単のために 3 層ニューラルネットワークを適用する．

まず，$w_{ij}^1(i=1,2,\cdots,I;j=1,2,\cdots,J)$ を入力層と中間層の結合係数，また $w_{jk}^2(j=1,2,\cdots,J;k=1,2,\cdots,K)$ は中間層と出力層の結合係数とする．さらに，$x_i(i=1,2,\cdots,I)$ は正規化された入力データを表し，本論文では，フォールト報告者により致命的であると判断されたフォールト数，特定の OS において発見されたフォールト数，システムの内部構造に習熟した修正者のフォールト修正数，システムの内部構造に習熟した発見者のフォールト発見数とした．ここで，入力層，中間層，出力層におけるユニットの数を，各々 $I$ 個，$J$ 個，および $K$ 個とする．また，各々の層のユニットを示すインデックスを $i$, $j$, および $k$ とする．ここで，各々の層のユニットの出力を $h_j, y_k$ とすると，

$$h_j = f\left(\sum_{i=1}^{I} w_{ij}^1 x_i\right), \tag{1}$$

$$y_k = f\left(\sum_{j=1}^{J} w_{jk}^2 h_j\right), \tag{2}$$

となる．但し，$f(\cdot)$ はシグモイド型関数であり，

$$f(x) = \frac{1}{1+e^{-\theta x}}, \tag{3}$$

として表される．ここで，$\theta$ はゲインと呼ばれる定数である．ネットワークの学習を行うために，誤差逆伝播法を用いる．ニューラルネットワークの出力層における値を $y_k(k=1,2,\cdots,K)$ とし，教師パターンを $d_k(k=1,2,\cdots,K)$ とすると，式 (2) の $y_k$ の評価は次式で与えられる．

$$E = \frac{1}{2}\sum_{k=1}^{K}(y_k - d_k)^2. \tag{4}$$

ここで，教師パターン $d_k(k=1,2,\cdots,K)$ には，各コンポーネントにおける累積発見フォールト数データの正規化された値を採用する．すなわち，各コンポーネントにおける累積フォールト発見数データに基づいて，各コンポーネントの重み係数とそれに影響を及ぼす要因の結合状態の特徴をニューラルネットワークの結合係数に蓄積させ，ある時点における各コンポーネントの重要度の推定・予測が可能なモデルを考える．式 (4) の条件のもとに，各結合係数が最急降下法にて以下のように決定される．

$$w_{jk}^2(\sigma+1) = w_{jk}^2(\sigma) + \epsilon(y_k - d_k)\cdot f'\left(\sum_{j=1}^{J} w_{jk}^2(\sigma)h_j\right)h_j, \tag{5}$$

$$w_{ij}^1(\sigma+1) = w_{ij}^1(\sigma) + \epsilon\sum_{k=1}^{K}(y_k - d_k)\cdot f'\left(\sum_{j=1}^{J} w_{jk}^2(\sigma)h_j\right)\cdot w_{ij}^1(\sigma) f'\left(\sum_{i=1}^{I} w_{ij}^1(\sigma)x_i\right)x_i. \tag{6}$$

ここで，$\sigma$ は更新のサイクル，$\epsilon$ は学習の係数を表す．式 (5) および式 (6) の更新により求められた $w_{ij}^1$ および $w_{jk}^2$ から，式 (1) および式 (2) により，ニューラルネットワークの出力層における値 $y_k(k=1,2,\cdots,K)$ を算出することができる．これにより，各コンポーネントに対する重みパラメータ $p_i(i=1,2,\cdots,K)$ を次の式により導出で

きる.

$$p_i = \frac{y_i}{\sum_{i=1}^{K} y_i}. \tag{7}$$

本論文では，ニューラルネットワークにおいて推定された各コンポーネントの重要度 $p_i$ を，システム全体の信頼性に対する各コンポーネントの影響率を表すものとする.

## 3 OSS に対する SRGM

### 3.1 一般化非同次ポアソン過程モデル

本論文で取り上げる OSS の動作環境は，様々なアプリケーションソフトウェアから影響を受け易く，従来のような同一組織内で開発され，単体で動作するソフトウェアシステムとは大きく環境が異なる．こうしたソフトウェア間の相互作用により，発見されるフォールト数も一定の値に収束することなく，将来的には増加し続けるものと考えられる.

本論文では，検出可能フォールト数が無限であると仮定された NHPP に基づく対数型ポアソン実行時間モデルを適用する．時間区間 $(0,t]$ で発見される総期待フォールト数を表す平均値関数 $\mu(t)$ は，

$$\mu(t) = \frac{1}{\theta - P}\ln[\lambda_0(\theta - P)t + 1] \quad (0 < \theta,\ 0 < \lambda_0,\ 0 < P < 1), \tag{8}$$

により与えられる．ここで，パラメータ $\lambda_0$ は初期故障強度，パラメータ $\theta$ はソフトウェア故障 1 個当りの故障強度の減少率を表す．また，パラメータ $P$ はシステム全体に及ぼすコンポーネントの影響率を表す．これは，各コンポーネントに対してニューラルネットワークを用いて推定されたパラメータ $y_i$ の平均値により表されるものと定義する [9].

### 3.2 一般化確率微分方程式モデル

本論文では，確率微分方程式から導出されたソフトウェア信頼度成長モデルを適用する．まず，時刻 $t = 0$ で OSS がリリースされ，任意の時刻 $t$ におけるソフトウェア内の発見フォールト数 $\{S(t), t \geq 0\}$ は以下の常微分方程式によって記述されるものと仮定する.

$$\frac{dS(t)}{dt} = \lambda(t)S(t). \tag{9}$$

ここで，$\lambda(t)(> 0)$ は時刻 $t$ におけるソフトウェア故障強度を表す.

OSS のフォールト報告では，フォールトの発見と同時にバグトラッキングシステムへのフォールト情報の登録が行われるというわけではなく，フォールト発見の前後に若干のタイムラグが生じた状態で登録が行われる場合が多い．このように，バグトラッキングシステム上へのフォールトの登録を考えた場合，OSS のフォールト発見事象は，常に不規則な状態であると考えられる．こうした不規則性を，標準化された Gauss 型白色雑音によって近似的に表現することを考える．さらに，OSS は常にバグフィックスやコンポーネントの加除が繰り返されており，ソフトウェア故障強度もそれに応じて変化するものと考えられる.

上記のことから，故障強度 $\lambda(t)$ に不規則性を導入すると，式 (9) は，

$$\frac{dS(t)}{dt} = \{\lambda(t) + \sigma\gamma(t)\}\, S(t), \tag{10}$$

となる．ここで，$\sigma$ $(> 0)$ は定数パラメータであり，$\gamma(t)$ は解過程の Markov 性を保証するために標準化された Gauss 型白色雑音である．さらに，$\lambda(t)$ は時刻 $t$ におけるソフトウェア故障強度関数を表す．式 (10) を以下の Itô 型 [10, 11] の確率微分方程式に拡張して考える [12].

$$dS(t) = \{\lambda(t) + \frac{1}{2}\sigma^2\}S(t)dt + \sigma S(t)d\omega(t). \tag{11}$$

式 (11) の確率微分方程式を，初期条件 $S(0)=v$ の下で，Itô の公式 [10, 11] を用いて変換すると，

$$S(t)=v\cdot\exp\left(\int_0^t\lambda(s)ds+\sigma W(t)\right), \tag{12}$$

となる．ここで，$v$ は以前のバージョンまでに発見されたフォールト数を表す．式 (12) で定義された $W(t)$ の性質は，次の通りである．

1. $W(t)$ は Gauss 過程である．
2. $W(t)$ の平均および分散は，それぞれ

$$\mathrm{E}[W(t)]=0,\mathrm{Var}[W(t)]=\sigma^2t, \tag{13}$$

   により与えられる．
3. $W(t)$ は定常独立増分をもつ．
4. $\Pr[W(0)=0]=1$.

本論文では，ソフトウェア故障強度関数を以下のように仮定する．

$$\int_0^t\lambda(s)ds=(1-\exp[-\alpha t]). \tag{14}$$

ここで，$\alpha$ はソフトウェア故障強度の加速係数を表す．式 (14) の強度関数は，その他にも S 字形などの形状を仮定することも可能である．しかしながら，OSS のフォールト報告の特徴として，バージョンアップ直後においては，フォールト報告数は指数関数的に増加するという経験的傾向があることから，これを反映するために指数形の強度関数を仮定した．さらに，モデルの構築および適用可能性に関する考察に重点を置くために，単純な構造をもつ式 (14) の強度関数について取り扱うこととする．

任意の時刻 $t$ における発見フォールト数の期待値 $\mathrm{E}[S(t)]$ および分散 $\mathrm{Var}[S(t)]$ は，ソフトウェア信頼性を評価する上で重要な尺度となる．これらは，Wiener 過程 $W(t)$ の密度関数が，

$$f(W(t))=\frac{1}{\sqrt{2\pi t}}\exp\left\{-\frac{W(t)^2}{2t}\right\}, \tag{15}$$

であることから，

$$\mathrm{E}[S(t)] \;=\; v\cdot\exp\left(\int_0^t\lambda(s)ds+\frac{\sigma^2}{2}t\right), \tag{16}$$

となる．

## 3.3 ソフトウェア信頼性評価尺度

上述した一般化非同次ポアソン過程モデル（NHPP model）および一般化確率微分方程式モデル（SDE model）から，種々のソフトウェア信頼性評価のための定量的尺度を導出できる．

瞬間フォールト発見率は強度関数により表すことができる．これは，単位時間当りに発見されるフォールト数として定義される．瞬間フォールト発見率は，一般化 NHPP モデルから以下のように導出できる．

$$\mu_d(t)=\frac{d\mu(t)}{dt}. \tag{17}$$

また，一般化 SDE モデルから，$S(t)$ の分散を次のように求めることができる．

$$\mathrm{Var}[S(t)]=\mathrm{E}[\{S(t)-\mathrm{E}[S(t)]\}^2]=v^2\exp\left(2\int_0^t\lambda(s)ds+\sigma^2t\right)\left\{\exp(\sigma^2t)-1\right\}. \tag{18}$$

平均ソフトウェア故障発生時間間隔（mean time between software failures: MTBF）は，ソフトウェア 故障の発生頻度を表すのに有益な尺度である．また，MTBF が大きな値を取ることは，それだけフォールトが発見し難く

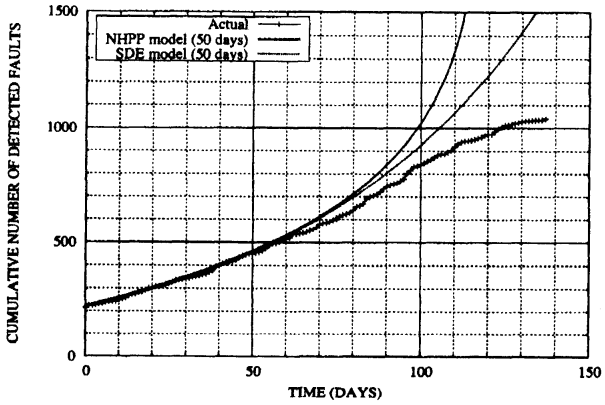


図 1： Firefox Web ブラウザに対する推定された累積フォールト発見数の期待値（50 日）.

なり，ソフトウェア信頼性が向上したと判断できることになる．任意の時刻 $t$ における瞬間 MTBF （instantaneous MTBF: $\mathrm{MTBF_I}$）および累積 MTBF（cumulative MTBF: $\mathrm{MTBF_C}$）は，一般化 NHPP モデルおよび一般化 SDE モデルから，以下のように導出できる．

任意の時刻 $t$ における瞬間的なフォールト発見間隔の平均を意味する瞬間 MTBF は，

$$MTBF_I(t) = \frac{1}{d\mu(t)/dt}, \tag{19}$$

$$MTBF_I(t) = \mathrm{E}\left[\frac{1}{dS(t)/dt}\right], \tag{20}$$

となる．

運用開始時点から考えたときの発見フォールト 1 個当りに要する発見時間の平均を意味する累積 MTBF は，

$$MTBF_C(t) = \frac{t}{\mu(t)}, \tag{21}$$

$$MTBF_C(t) = \mathrm{E}\left[\frac{t}{S(t)}\right], \tag{22}$$

により表すことができる．

## 4 実測データに適用した予測精度の検証

3 種類の OSS における信頼性評価結果を示す．評価版において十分な信頼性を確認することは，正式版リリース後のユーザに対する信頼や人気に大きくかかわるだけでなく，OSS の保守コストや保守労力の増大に関係することから，リリース候補版の評価は正式版リリースへ向けての重要な段階となる．2 種類の SRGM に基づく信頼性評価法により推定された各 OSS に対する信頼性評価結果として，推定された累積フォールト発見数の期待値の推定値を図 1 から図 6 に示す．本論文では，各 OSS のリリース候補版におけるデータを使用した．特に，評価版リリースから特定の時刻までのデータを使用し，それ以降の予測結果の比較を行っている．

まず，図 1 および図 2 は，Firefox Web ブラウザ [13] における推定された累積フォールト発見数の期待値の推定値を表しており，図 1 は評価版のリリース以降 50 日目までのデータを使用して推定された結果である．同様に，図 2 は，評価版のリリース以降 100 日目までのデータを使用して推定し，それ以降の比較を行った結果を示す．また，Apache HTTP サーバ [14] における推定された累積フォールト発見数の期待値の推定結果を図 3 および図 4 に示す．さらに，Fedora Core Linux[15] における推定された累積フォールト発見数の期待値の推定結果を図 5 および図 6 に示す．上記の結果から，NHPP モデルよりも SDE モデルの適合性が良好であることが確認でき，推定に使用されるデータ数が多くなるほど誤差も小さくなることが分かった．

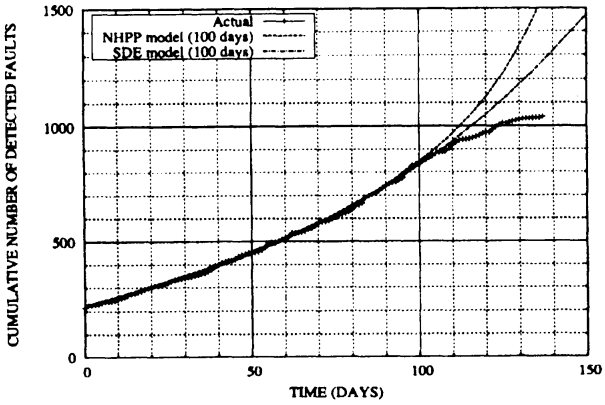


図 2： Firefox Web ブラウザに対する推定された累積フォールト発見数の期待値（100 日）.

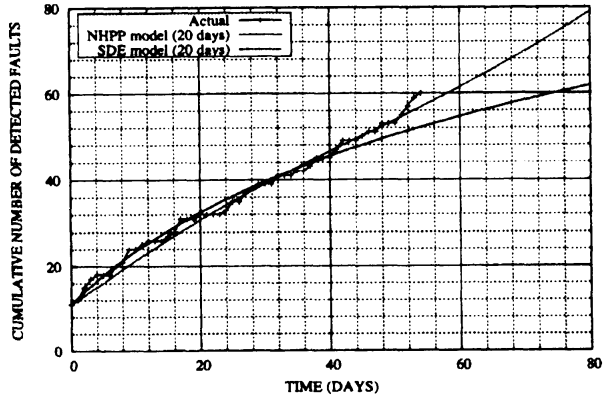


図 3： Apache HTTP サーバに対する推定された累積フォールト発見数の期待値（20 日）.

## 5 おわりに

本論文では，オープンソースプロジェクトの下で開発されている OSS に対する信頼性評価法を提案した．特に，OSS の開発形態を考慮するために非同次ポアソン過程および確率微分方程式に基づいた SRGM に基づく信頼性評価結果として，実際のオープンソースプロジェクトにおけるバグトラッキングシステムから採取されたフォールトデータに対する数値例を示した．さらに，各 SRGM の予測精度の比較を行った．

また，アプリケーション，サーバ，OS における 3 種類の OSS のフォールトデータから各モデルの予測精度の比較を行った結果から，NHPP モデルよりも SDE モデルの適合性が良好であることが確認でき，推定に使用されるデータ数が多くなるほど誤差も小さくなることが分かった．これまで，OSS の開発工程ではソフトウェアの信頼性を動的かつ定量的に評価するという試みが行われていなかったことから，本論文において予測精度の比較を行った SRGM をオープンソースプロジェクトに適用することによって，より高品質な OSS の開発に結びつくものと考える．

## 謝辞

本論文の一部は，文部科学省科学研究費基盤研究 (C)（課題番号 18510124）の援助を受けたことを付記する．

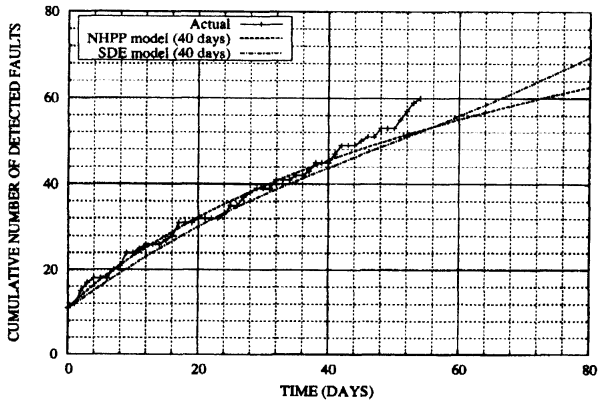


図 4： Apache HTTP サーバに対する推定された累積フォールト発見数の期待値（40 日）.

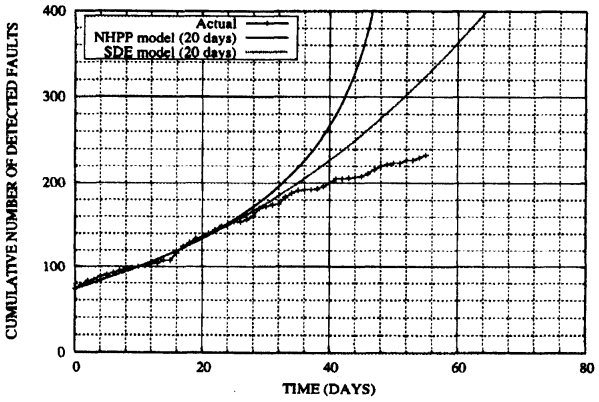


図 5： Fedora Core Linux に対する推定された累積フォールト発見数の期待値（20 日）.

## 参考文献

[1] E-Soft Inc., Internet Research Reports, http://www.securityspace.com/s_survey/data/

[2] ソフトウェア情報センター研究会報告書, オープンソースソフトウエアの利用状況調査／導入検討ガイドラインの公表について, 東京, 2004.

[3] 山田 茂, ソフトウェア信頼性モデルー基礎と応用ー, 日科技連出版社, 東京, 1994.

[4] A. MacCormack, J. Rusnak, and C.Y. Baldwin, "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code," *Informs Journal of Management Science*, vol. 52, no. 7, pp. 1015–1030, 2006.

[5] G. Kuk, "Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List," *Informs Journal of Management Science*, vol. 52, no. 7, pp. 1031–1042, 2006.

[6] Y. Zhoum, J. Davis, "Open Source Software Reliability Model: An Empirical Approach," *Proceedings of the workshop on Open Source Software Engineering (WOSSE)*, vol. 30, no. 4, 2005, pp. 67–72.

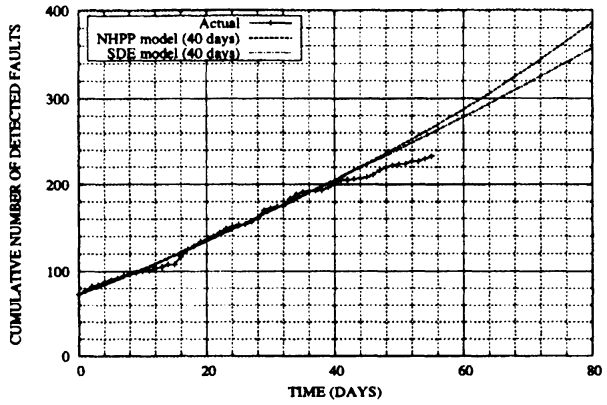


図 6： Fedora Core Linux に対する推定された累積フォールト発見数の期待値（40 日）.

[7] P. Li, M. Shaw, J. Herbsleb, B. Ray and P. Santhanam, "Empirical Evaluation of Defect Projection Models for Widely-deployed Production Software Systems," *Proceedings of 12th International Symposium on the Foundations of Software Engineering (FSE-12)*, 2004, pp. 263–272.

[8] E. D. Karnin, "A Simple Procedure for Pruning Back-propagation Trained Neural Networks," *IEEE Trans. Neural Networks.*, vol. 1, pp. 239–242, 1990.

[9] Y. Tamura and S. Yamada, "A Method of User-oriented Reliability Assessment for Open Source Software and Its Applications," *Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics*, Taipei, Taiwan, Oct. 8–11, 2006, pp. 2185–2190.

[10] L. Arnold, *Stochastic Differential Equations–Theory and Applications*, New York, John Wiley & Sons, 1974.

[11] E. Wong, *Stochastic Processes in Information and Systems*, New York, McGraw–Hill, 1971.

[12] S. Yamada, M. Kimura, H. Tanaka, and S. Osaki, "Software Reliability Measurement and Assessment with Stochastic Differential Equations," *IEICE Trans. Fundamentals*, vol. E77-A, no. 1, pp. 109-116, Jan. 1994.

[13] Mozilla, Firefox, http://www.mozilla.com/en-US/firefox/

[14] E-Soft Inc., Internet Research Reports, http://www.securityspace.com/s_survey/data/

[15] Fedora Project, sponsored by Red Hat, http://fedora.redhat.com/